Using a Real-Time, QoS-based ORB to Intelligently Manage Communications Bandwidth in a Multi-Protocol Environment

Bill Beckwith

Objective Interface Systems, Inc.

OMG Embedded Workshop

The Nature of CORBA

- ORBs abstract away network semantics
  - Application design is much less constrained by the communications infrastructure
  - *But* the application developer has little direct control over the communication paths and parameters used
- Thus, ORBs make it difficult to
  - Control communication channels
  - Much less manage communications bandwidth
  - Much less schedule the communications channels

What is QoS?

- QoS = Quality of Service
- QoS means different things to different people
- QoS definition for this presentation
  - The parameters offered by a communications transport for affecting the characteristics of one communication channel vs. another
  - *Not* a scheduling parameter (see Dynamic Scheduling)

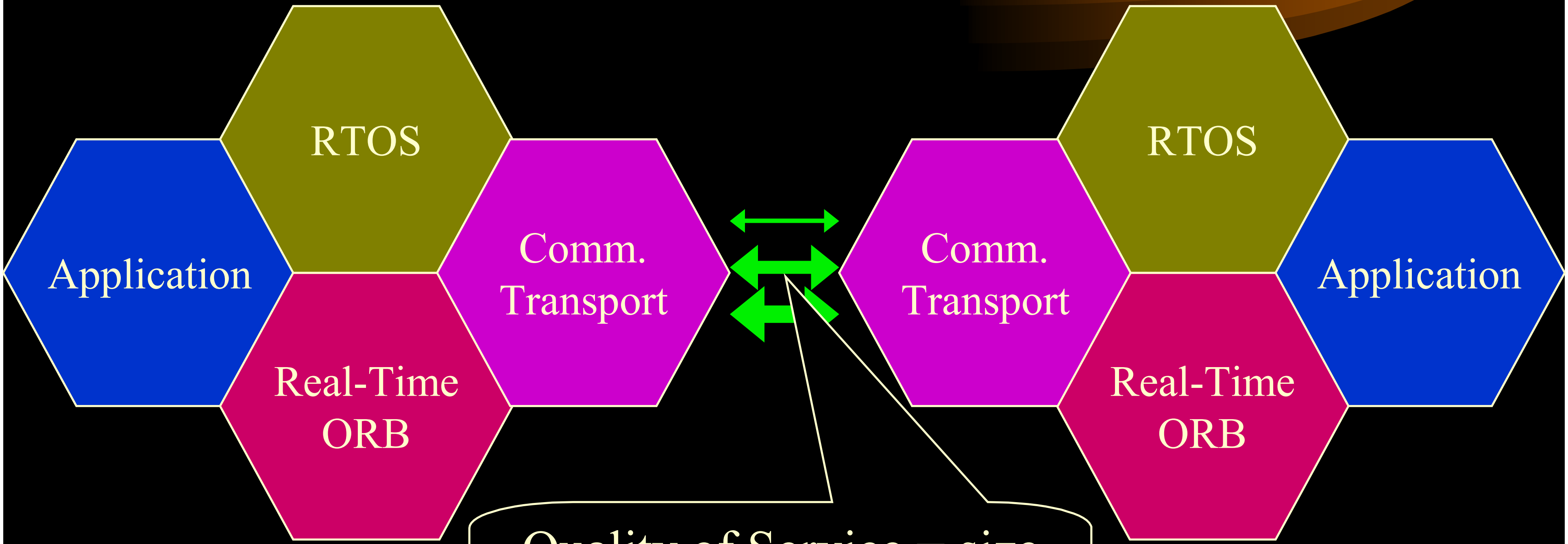
Why QoS?

- Facilitates transports that can
  - maintain priority
  - distribute bandwidth
  - guarantee jitter
  - bound latency
  - etc.
- A needed general abstraction of what transports offer

Most ORBs Ignore QoS

- Middleware and ORBs have traditionally ignored transport QoS
- Middleware can’t tell transport to favor one connection over another
- Middleware must live with either default settings or *worse* TCP/IP layered on top of transport

Quality of Service in a Real-time CORBA Application



Quality of Service = size and characteristics of pipe

Processor Scheduling

- Most real-time theory ignores the unsolvable problem
  - Scheduling multi-processor, multi-node, multi-transport distributed systems
  - Instead focuses on scheduling the use of a processor in a system
  - The harder problem exists in many real-time systems and isn’t going away

End-to-end Latency and Jitter

- Crucial to identify and isolate the sources of latency and jitter
  - Key to understanding the benefits of QoS
- Contributors to latency and jitter
  - Application
  - Real-time ORB
  - Replaceable transport plug-in
  - Communications transport (protocol stack and media)
  - Operating system
  - Higher priority activities

Application Latency and Jitter

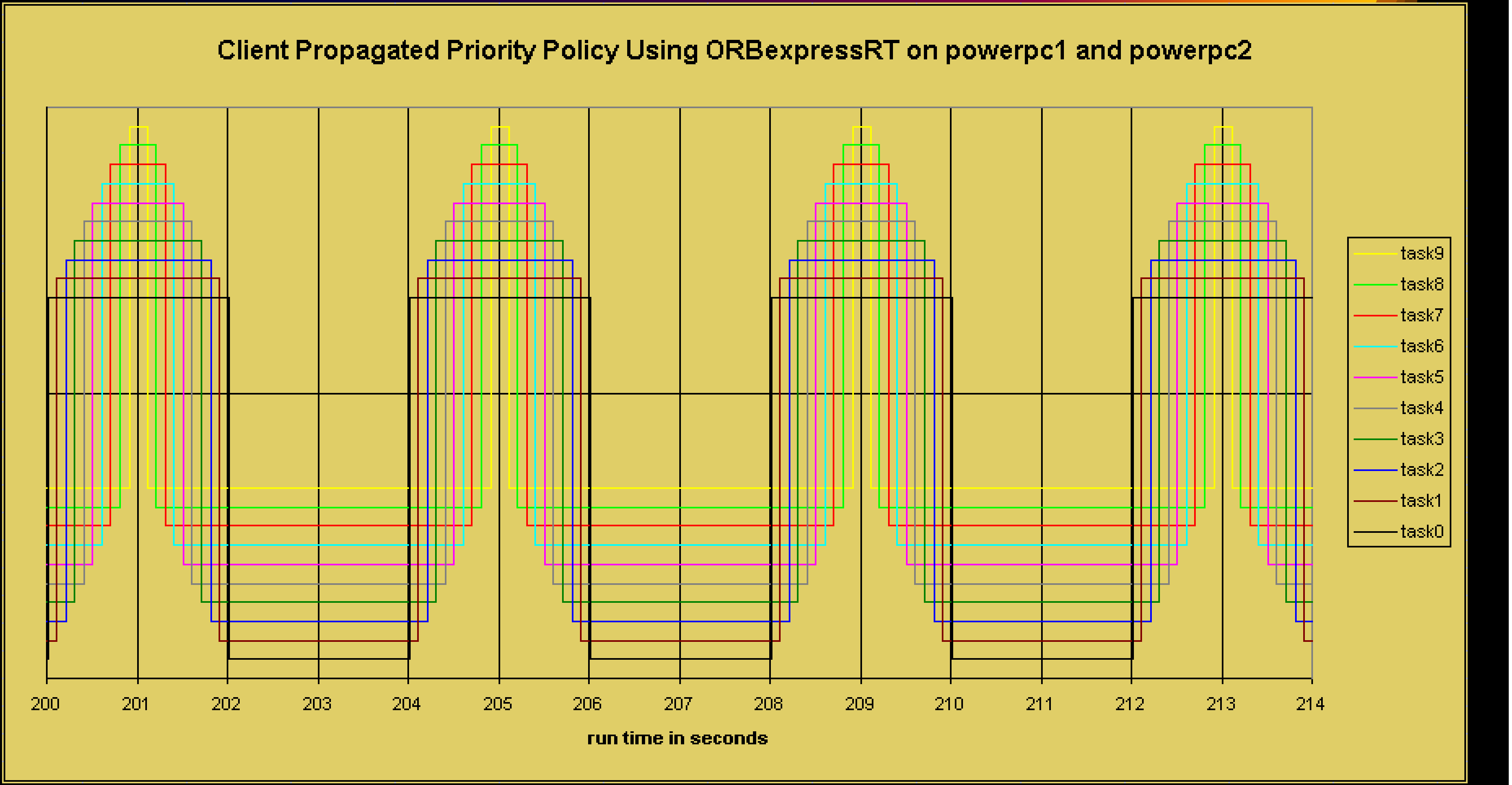
- Mostly within the control of the application engineer
- ORB should facilitate application’s control of scheduling requirements (i.e. Dynamic Scheduling for CORBA)
- ORB should aid developer in avoiding priority inversions and priority deadlock

Real-time ORB Latency and Jitter

- A virtue or vice of the ORB implementation
- Depends on the quality of ORB implementation
- Is well within the control of the real-time ORB implementer
- A well implemented real-time ORB should add very little or no jitter
- Real-time ORBs exist that add very little latency
- Real-time ORBs exist without priority inversions

Real-Time ORB Priority Testing

Boeing Second Phase Test Results – Nov 20, 2000



Client Propagated Priority Policy Using ORBexpressRT on powerpc1 and powerpc2

Replaceable Transport Plug-in Latency and Jitter

- Also may be developed by application engineer
- Can introduce additional latency and jitter if poorly designed
- A properly engineered transport *shouldn't* introduce to jitter

Communications Transport Latency and Jitter

- Frequently *the* significant source of latency and jitter
- More advanced transports offer hope
  - Hope is spelled QoS
  - Some are bandwidth oriented
  - Others are priority oriented

Operating System Latency and Jitter

- Easy to test for, hard to isolate
- Caused by:
  - Poor scheduling
  - Bad algorithms in support libraries
    - Watch out for printf()!
  - Poor priority management in support libraries
    - No priority inheritance in O/S mutexes
    - Running protocol stacks at lower priorities
- Only use RTOSes for real-time systems

Jitter Caused by High Priority or Other Activities

- May or may not be under application developers control
- Where possible, remove spurious interrupts
  - If your invocations across IEEE 1394 are showing large jitter:

    Unplug the Ethernet cable from your board!
- Bus contention by I/O devices, etc.

Various Communications Transports

- Ethernet
  - Random hardware interrupts
  - Variable workload caused by each interrupt
  - Hubs add less latency, more jitter
  - Switches add latency, less jitter
  - No QoS with standard Ethernet
  - Various technologies for switched Ethernet QoS

Various Communications Transports (cont.)

- TCP/IP
  - – Fine with Ethernet if you have no low latency, bounded latency or jitter requirements ;-)
  - – Needlessly duplicates reliability that may be available as a QoS parameter in a lower level transport (e.g. ATM virtual circuits)
  - – Many protocol stacks turn Ethernet’s random interrupts into random workloads

Various Communications Transports (cont.)

- ATM
  - Random hardware interrupts
  - Variable workload caused by each interrupt
  - Complex set of QoS parameters allow some throttling of data flow
- IEEE 1394/FireWire
  - Periodic interrupts
  - Variable workload caused by each interrupt
  - QoS is isochronous bandwidth allocation

Various Communications Transports (cont.)

- Reflective memory
  - *No* hardware interrupts
  - Contention for media is bounded
  - Highly predictable
  - Little available QoS parameters
- Switched Fabrics (RACEway, RapidIO, Infiniband)
  - *No* hardware interrupts
  - No media contention (N x N traffic)
  - QoS is priority for resolving contention if N is exceeded

A QoS Cognizant ORB Implementation

- ORB*express* RT allows full control of QoS on user defined transports
- ORB*express* RT blends user control of
  - Processor scheduling and
  - Communications channel scheduling.
- Supports wide range of optimization criteria
  - Bounded latency (hard real-time) systems
  - High throughput, cost constrained soft real-time systems

Managing Communication Channels

- ORB*express* RT introduced Real-Time QoS Interceptors™
  - Give full control to the developer to manage the ORBs use of QoS and each communication channel
  - Default interceptor conforms to Real-time CORBA 1.0 specification

User Replaceable Transports with QoS

- ORB*express* GT was the first ORB with user replaceable transports with or in lieu of TCP/IP
- ORB*express* RT evolved this architecture for replaceable transports by allowing application developers to define new classes of QoS as supported by their plug-in transport

Separating Application Design from Scheduling

- ORBs are a golden opportunity for managing application scheduling
- Requires access to and control of
  - Threads
  - Resource management
  - Communication channels
- As a result ORB*express* RT allows developers to separate
  - application design from
  - application scheduling

Portability Across Communication Protocols

- Combining
  - Management of processor availability and
  - Communications QoS
- Yields improved ability to port an application among differing communications protocols
- Porting then involves
  - Easy porting of source code
  - Completely new scheduling analysis